

Optimizing Index based on Text Categorization by Graph based Version of KNN

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the graph based KNN version for automating the index optimization by introducing the text classification. In the previous work, we proposed the graph based KNN version as an approach to the index optimization, but must tag the input text with its domain, manually. In this research, we use the graph-based version which receives a graph, instead of a numerical vector for both the text classification and the keyword extraction. In the proposed system, only a text is given as the input without presenting the domain, it is assigned to the text by the text classification, automatically, and each of in the text is classified into one of the three categories by the classifier which corresponds to the domain. The goal of this research is to automate fully the index optimization by combining it with the text classification, and to improve the performance of both tasks, using the graph based KNN algorithm.

I. INTRODUCTION

The index optimization is the task for maximizing the information retrieval performance by optimizing the text index. Both the index expansion and the index filtering are used for implementing the index optimization, it is necessary to define words with the three classes, expansion, inclusion, and removal, and the task is converted into a classification task. The process of index optimization is to index a text which is given as the input into a list of words, classify each word into one of the three categories, and treat words differently depending on their classes. To the words which are classified into expansion, associative words are added from external sources, the words which are classified into inclusion are taken as indexes, and the words which are classified into removal are excluded from the indexes. Because the classification which is mapped from the index optimization is a domain dependent task where a same entity may be classified differently depending on the domain, it is necessary to present the domain as a text tag for doing the index optimization.

This research is motivated by the requirement of knowing the domain in advance for executing the index optimization. It is set as an independent classification task within each domain. It is necessary to selecting a classifier which corresponds to the domain for executing the index optimization. It is possible to assign the domain automatically to the text through the text classification. The index optimization which is covered in this research relates to the text classification.

We apply the graph based KNN algorithm to the index optimization which relates to the text classification. We define the similarity metric between graphs based on one between edges and modify the KNN algorithm into the version which processes graphs directly. The graph based KNN algorithm is adopted for implementing the text classification which automatically labels the input text with its own domain. The text is indexed into words, and the modified version is adopted for classifying words into one of the three categories. In this research, we propose the graph based KNN algorithm as the approach to both the index optimization and the text classification.

Let us mention some benefits from this research. Encoding words or texts into graphs is the solution to the problems in encoding them into numerical vectors. The performance of the index optimization and the text classification is improved by adopting the graph based KNN algorithm as the approach to both tasks. The manual assignment of a domain to an input text is avoided for using the index optimization program. In this research, we automate the process of nominating a classifier which corresponds to a domain as the system upgrade.

Let us mention some remaining tasks as the further discussion on this research. We define more operations on graphs as well as the similarity metric. We consider representing words or texts into compound structured forms, each of which consists of more than one structured form. Other machine learning algorithms and deep learning algorithms may be modified into graph-based versions. The index optimization system may be implemented as a real program or a library module by adopting the graph based KNN algorithm.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the graph as a word representation and the similarity between graphs. In representing a word into a graph, a vertex indicates a text, and an edge indicates a similarity between texts. The similarity between edges is computed, before computing the similarity between graphs, viewing a graph as a set of edges. The similarity between graphs is computed by averaging over the similarities of all possible pairs of edges. In this section,

we describe the process of encoding a word into a graph and the process of computing the similarity between graphs.

The process of encoding words into graphs is illustrated in Figure 1. In the vertex definition, from a text collection, texts which are relevant to the word are retrieved as the vertices. In the edge definition, the similarities among the retrieved texts are computed as edges. In the edge selection, the edges with their higher similarities are selected for representing a graph. Refer to [2] for studying the encoding process in more detail.

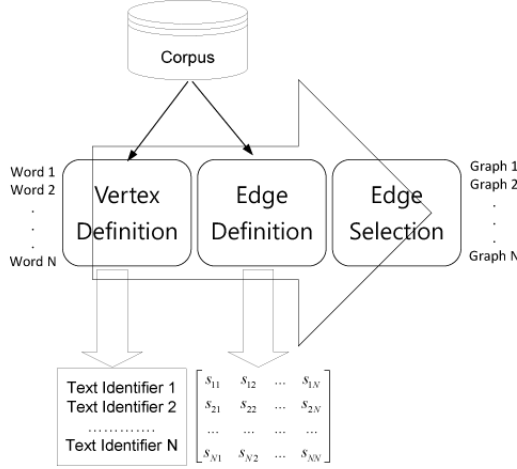


Figure 1. Process of Encoding Words into Graphs

The three cases of computing the similarity between two edges are illustrated in Figure 2. Each weight which is associated with its own edge is assumed to be a normalized value between zero and one, and if both nodes are identical to each other, the average over two weights is the similarity between two edges. If either of the two nodes is identical to each other, the product of two weights is the similarity between two edges. If neither of two nodes is identical, zero is the similarity between them. The similarity between two edges is always a normalized value between zero and one.

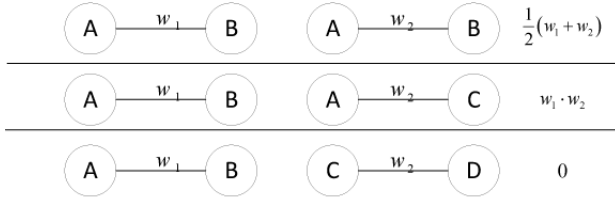


Figure 2. Three Cases of Comparing Two Edges with Each Other

Let us mention the process of computing the similarity between graphs as a semantic similarity between words. The two graphs which represent words are notated by edge sets, $G_1 = \{e_{11}, e_{12}, \dots, e_{1|G_1|}\}$ and $G_2 = \{e_{21}, e_{22}, \dots, e_{2|G_2|}\}$. Two edges, $e_{1i} \in G_1$ and $e_{2j} \in G_2$, are associated with their own weights, w_{1i} and w_{2j} , and

the similarity between two edges, e_{1i} and e_{2j} by equation (1), (2), or (3), depending on the cases which are mentioned above,

$$\text{sim}(e_{1i}, e_{2j}) = \frac{1}{2}(w_{1i} + w_{2j}) \quad (1)$$

$$\text{sim}(e_{1i}, e_{2j}) = w_{1i} \times w_{2j} \quad (2)$$

$$\text{sim}(e_{1i}, e_{2j}) = 0 \quad (3)$$

The similarity between two graphs is computed by equation (4),

$$\text{sim}(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} \text{sim}(e_{1i}, e_{2j}). \quad (4)$$

The value which is generated from equation (4), is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(G_1, G_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding a word into a graph and computing the similarity between graphs. A word is encoded into a graph by the vertex definition, the edge definition, and the edge weighting. The three cases are considered for computing the similarity between edges. The similarity between graphs is computed by equation (4). In the future research, we consider representing a word into multiple graphs.

B. Graph based KNN Algorithm

This section is concerned with the graph based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 [2]. The similarity metric is used for computing the similarity between a novice graph and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into graphs, and they are notated by a set, $Tr = \{G_1, G_2, \dots, G_N\}$. A novice word is encoded into a graph notated by G . Its similarities with the graphs which represent the sample words are computed by equation (4), as $\text{sim}(G, G_1), \text{sim}(G, G_2), \dots, \text{sim}(G, G_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice

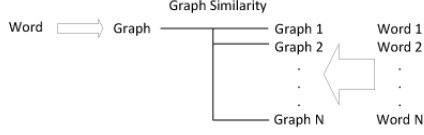


Figure 3. Similarities of Novice Input with Training Examples

example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, G), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

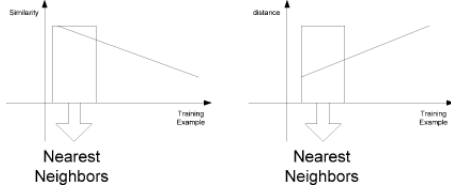


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(G_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, G , is decided by equation (8),

$$\text{label}(G) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

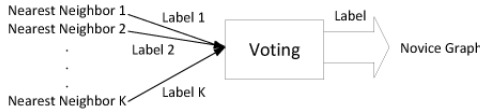


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the graph based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the graphs based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into graphs for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

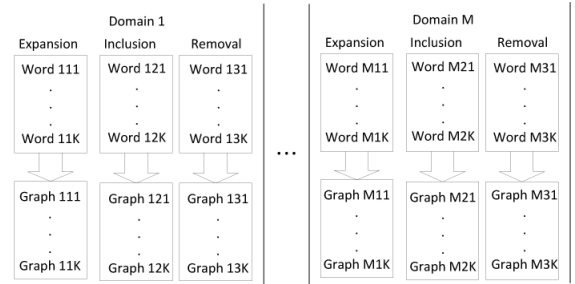


Figure 6. Preparation of Training Examples

The entire architecture of the proposed index optimization system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. In the encoding module, the sample words in the groups, expansion, inclusion, and removal and ones which are indexed from the text are encoded into graphs. In the similarity computation module and the voting module, the words which are indexed from the text are classified into one of the three categories. The words which are classified into removal are discarded in this system.

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into graphs. An input text is indexed into a list of words, and they are also encoded into graphs. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into

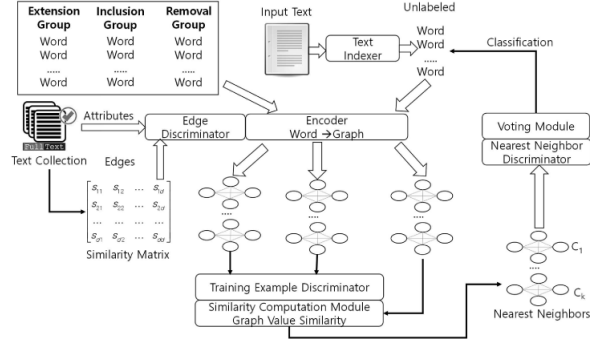


Figure 7. System Architecture

removal are excluded.

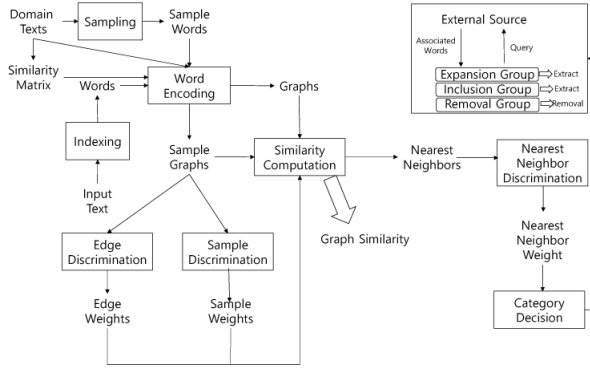


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The index optimization is defined as a multiple classification task where each word is classified into expansion, inclusion, or removal. The words are encoded into graphs instead of numerical vectors, and they are classified directly. The words which are classified into removal are excluded from index of the input text. In implementing the system as its complete version, we need to add the module which retrieve more words which are relevant to ones which are labeled with expansion.

D. Connection with Text Classification

This section is concerned with the connection of the index optimization with the text classification. In the previous section, we mentioned the index optimization system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the index optimization system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into graphs by the process which is described in Section II-A. The similarity computation module is for computing the similarities between graphs which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

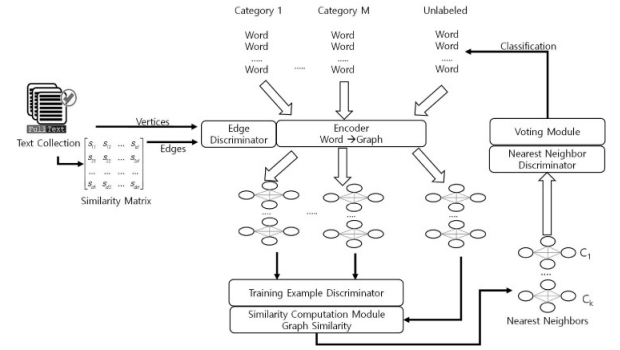


Figure 9. Text Categorization System

The connection of the index optimization with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the index optimization system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the index optimization, automatically.

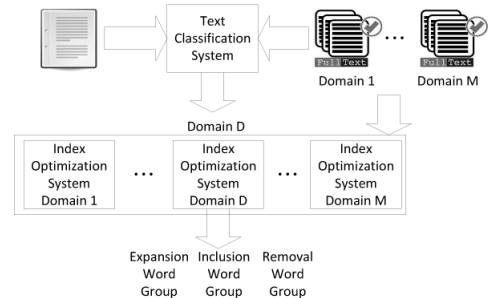


Figure 10. Index Optimization System connected with Text categorization

The process of executing the index optimization, connecting it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its own domain are prepared, and the sample words each of which

is labeled with one of the three categories are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain. The novice text is indexed into a list of words, and each word is classified into one of the three categories. The labeled words are the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

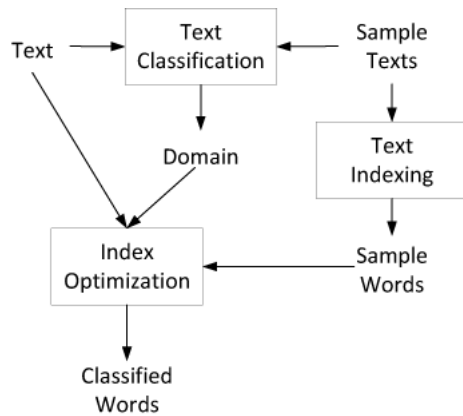


Figure 11. System Flow of Index Optimization connected with Text Categorization

Let us make some remarks on the connection of the index optimization with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the index optimization is to classify words into expansion, inclusion, or removal. In the execution flow, the input text is classified into a domain, it is indexed into a list of words, and each word is classified into one of the three categories. We consider connecting the text classification as the main task the index optimization as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Graph based KNN for Optimizing Index of News Articles", 53-62, Journal of Multimedia Information System, 3, 2016.
- [2] T. Jo, "Comparing Graph based K Nearest Neighbor with Traditional Version in Word Categorization in NewsPage.com, 12-18, International Journal of Advanced Social Sciences, 1, 2018.